Learning with Multiplicative Perturbations

Xiulong Yang (xyang22), Jonathan Shihao Ji (sji), Georgia State University @gsu.edu

Adversarial Examples

Deep neural networks (DNNs) have demonstrated remarkable success in solving complex prediction tasks. However, recent studies show that they are particularly vulnerable to adversarial attacks in the form of small perturbations to inputs that lead DNNs to predict incorrect outputs.

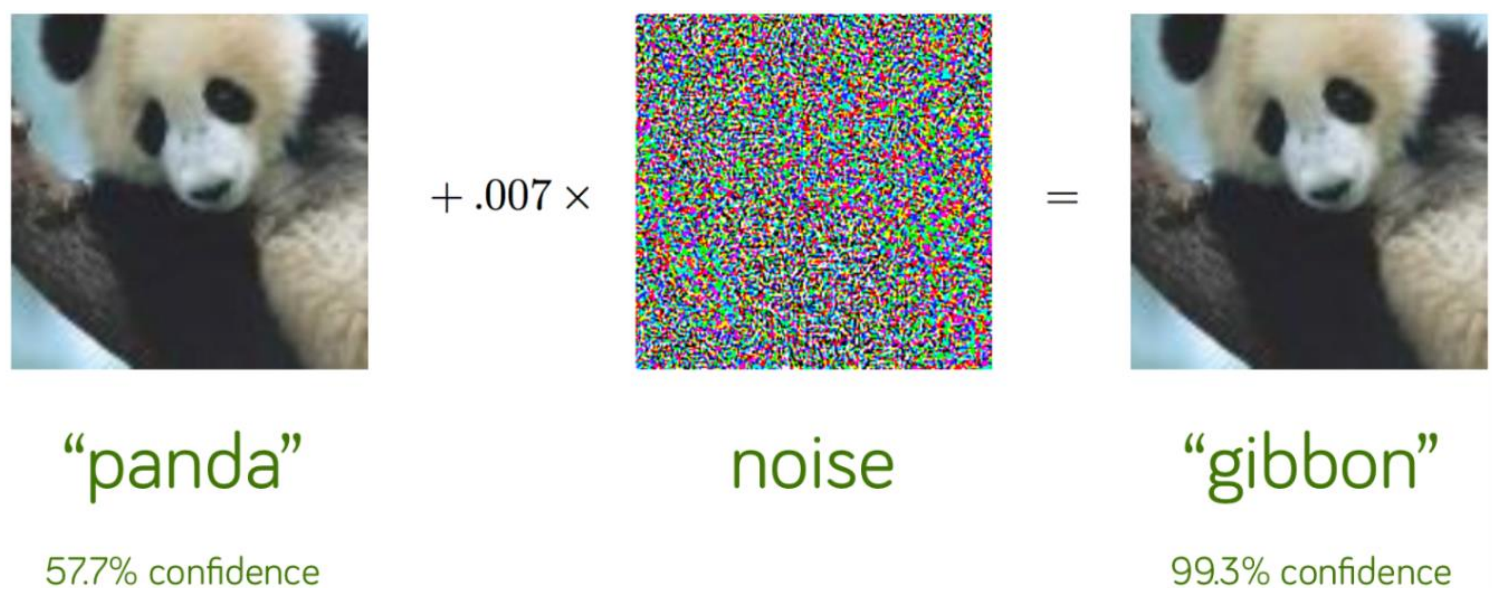


Figure 1. Adversarial Example[1].

Adversarial Training: AT and VAT

Several studies have found that the performance of DNNs can be improved significantly by enforcing the prediction consistency of DNNs in response to original inputs and their perturbed versions.

To improve the robustness of DNNs, researchers propose different approaches to regularize the training of DNNs by augmenting the training set with adversarial examples, such as AT[1] only for supervised learning, and VAT[2] for both supervised learning and semi-supervised learning.

AT solves the following constrained optimization problem:

$$\mathcal{L}_{\mathrm{AT}}(\boldsymbol{x}_l, y_l, \boldsymbol{r}_{\mathrm{adv}}, \boldsymbol{\theta}) = D\left[h(y_l|\boldsymbol{x}_l), p(y|\boldsymbol{x}_l + \boldsymbol{r}_{\mathrm{adv}}, \boldsymbol{\theta})\right]$$
$$\text{with } \boldsymbol{r}_{\mathrm{adv}} = \underset{\boldsymbol{r};\|\boldsymbol{r}\|\le\epsilon}{\arg\max} D\left[h(y_l|\boldsymbol{x}_l), p(y|\boldsymbol{x}_l + \boldsymbol{r}, \boldsymbol{\theta})\right],$$

VAT deals with a slightly different constrained optimization problem:

$$\mathcal{L}_{\mathrm{VAT}}(\boldsymbol{x}_*, \boldsymbol{r}_{\mathrm{adv}}, \boldsymbol{\theta}) = D\left[p(y|\boldsymbol{x}_*, \boldsymbol{\theta}), p(y|\boldsymbol{x}_* + \boldsymbol{r}_{\mathrm{adv}}, \boldsymbol{\theta})\right]$$
$$\text{with } \boldsymbol{r}_{\mathrm{adv}} = \underset{\boldsymbol{r};\|\boldsymbol{r}\|_2\le\epsilon}{\arg\max} D\left[p(y|\boldsymbol{x}_*, \boldsymbol{\theta}), p(y|\boldsymbol{x}_* + \boldsymbol{r}, \boldsymbol{\theta})\right],$$

However, the perturbations exploited by AT and VAT are additive in the sense that these perturbations are added pixel-wise to input examples.

Reference

1. I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples". In International Conference on Learning Representations(ICLR), 2015.
2. T. Miyato, S.-i. Maeda, M. Koyama, S. Ishii. "Virtual Adversarial Training a Regularization Method for Supervised and Semi-Supervised Learning". IEEE transactions on PAMI, 2018.
3. T. Bird, J. Kunze, D. Barber, "Stochastic variational optimization". arXiv: 1809.04855, 2018.
4. C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through L_0 regularization," in International Conference on Learning Representations (ICLR), 2018.

Multiplicative Perturbations

We propose a new type of adversarial perturbations that is multiplicative:

$$x_{adv} = x + r_{adv} \qquad x_{xadv} = x \odot z$$

With the new perturbations, we derive the new loss functions for xAT as

$$\mathcal{L}_{\mathrm{xAT}}(\boldsymbol{x}, \boldsymbol{z}_{\mathrm{xadv}}, \boldsymbol{\theta}) = D\left[h(y|\boldsymbol{x}, \boldsymbol{\theta}), p(y|\boldsymbol{x} \odot \boldsymbol{z}_{\mathrm{xadv}}, \boldsymbol{\theta})\right]$$
$$\text{with } \boldsymbol{z}_{\mathrm{xadv}} = \underset{\boldsymbol{z}}{\arg\max} D\left[h(y|\boldsymbol{x}, \boldsymbol{\theta}), p(y|\boldsymbol{x} \odot \boldsymbol{z}, \boldsymbol{\theta})\right]$$

And xVAT:

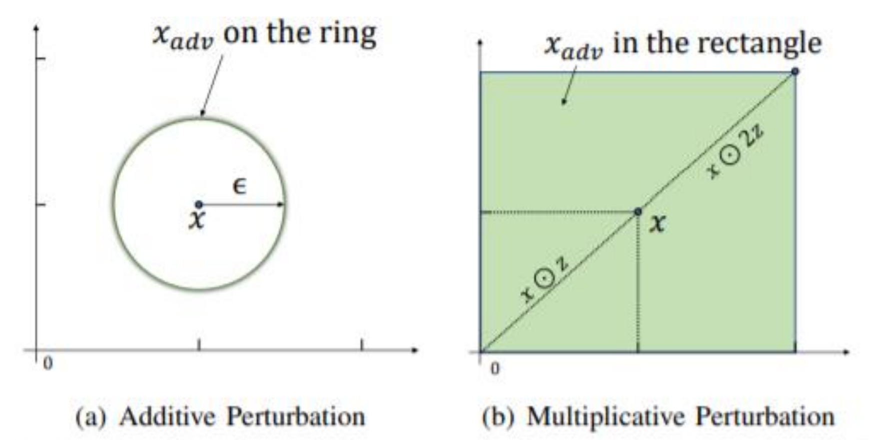
$$\mathcal{L}_{\mathrm{xVAT}}(\boldsymbol{x}, \boldsymbol{z}_{\mathrm{xadv}}, \boldsymbol{\theta}) = D\left[p(y|\boldsymbol{x}, \boldsymbol{\theta}), p(y|\boldsymbol{x} \odot \boldsymbol{z}_{\mathrm{xadv}}, \boldsymbol{\theta})\right]$$
$$\text{with } \boldsymbol{z}_{\mathrm{xadv}} = \underset{\boldsymbol{z}}{\arg\max} D\left[p(y|\boldsymbol{x}, \boldsymbol{\theta}), p(y|\boldsymbol{x} \odot \boldsymbol{z}, \boldsymbol{\theta})\right]$$

Compared to the additive perturbations exploited by AT and VAT, the multiplicative perturbations are:
(1) More perceptible (2) More interpretable

We use the L_0-norm of z to regularize the learning as Eq.(9)

$$\begin{aligned} \boldsymbol{z}_{\mathrm{xadv}} &= \underset{\boldsymbol{z}}{\arg\max}\, \Delta D(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{\theta}) + \lambda\|\boldsymbol{z}\|_0 \\ &= \underset{\boldsymbol{z}}{\arg\max}\, \Delta D(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{\theta}) + \lambda\sum_{j=1}^{P} 1_{[z^j \neq 0]} \end{aligned} \tag{9}$$

However, the discrete essence of z makes it undifferentiable.



(a) Additive Perturbation (b) Multiplicative Perturbation

Fig. 3. The effect of ϵ on different perturbations. (a) shows that the additive perturbations are on the surface of a ball with the radius ϵ. (b) demonstrates that our multiplicative perturbations are distributed within the rectangle.

Optimization and Efficient Computing

To address the undifferentiable issue in Eq.(9), We adopt Stochastic Variational Optimization[3] and the Hard Concrete Gradient Estimator[4] techniques to optimize Eq.(12):

$$\log \boldsymbol{\alpha}_{\mathrm{xadv}} = \underset{\log \boldsymbol{\alpha}}{\arg\max} \mathbb{E}_{\boldsymbol{u}\sim\mathcal{U}(0,1)}\left[\Delta D(g(f(\log\boldsymbol{\alpha}, \boldsymbol{u})), \boldsymbol{x}, \boldsymbol{\theta})\right] + \lambda\sum_{j=1}^{P}\sigma\left(\log\alpha^j - \beta\log\frac{-\gamma}{\zeta}\right) \tag{12}$$

with

$$f(\log\boldsymbol{\alpha}, \boldsymbol{u}) = \sigma\left((\log\boldsymbol{u} - \log(1-\boldsymbol{u}) + \log\boldsymbol{\alpha})/\beta\right)(\zeta-\gamma)+\gamma,$$
$$g(\cdot) = \min(1, \max(0, \cdot)),$$

Then we generate the mask by sampling:

$$\hat{\boldsymbol{z}}_{\mathrm{xadv}} = g(f(\log\boldsymbol{\alpha}_{\mathrm{xadv}}, \boldsymbol{u})), \qquad \boldsymbol{u} \sim \mathcal{U}(0,1).$$

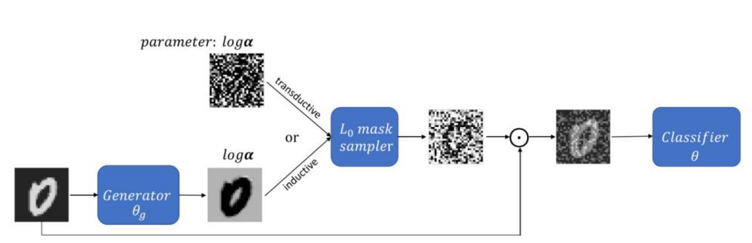
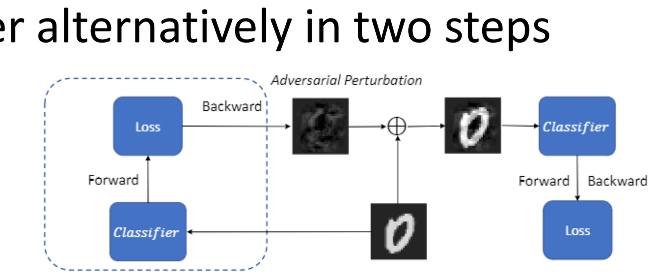


Fig. 2. The pipeline of the transductive and inductive implementations of multiplicative adversarial training.

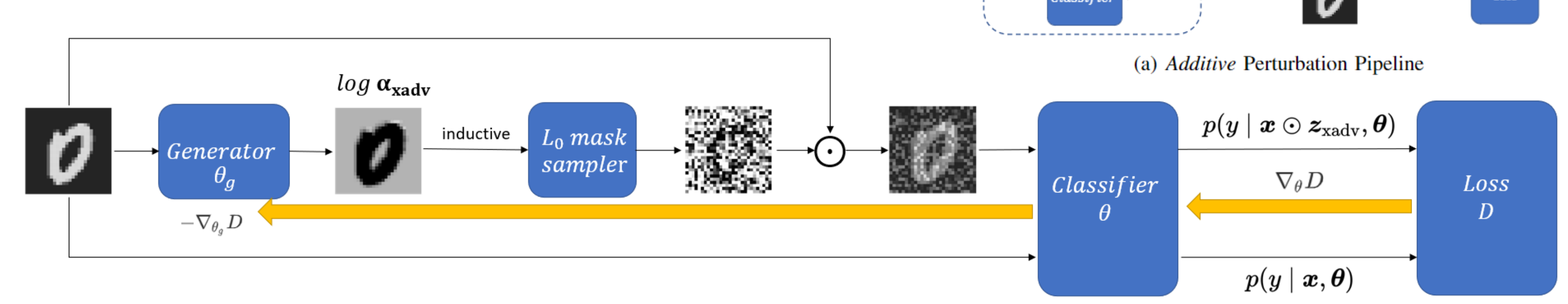
The optimization can be implemented transductively or inductively as Fig.2.

Both AT and VAT resort to optimizing additive perturbations and classifier parameter alternatively in two steps as below.



(a) Additive Perturbation Pipeline

And our xAT/xVAT can update them simultaneously in one step.



Compare with other methods

We evaluate the performance of xAT and xVAT in semi-supervised learning and supervised learning on MNIST, SVHN, CIFAR- 10 and CIFAR-100. And it demonstrates that xAT/xVAT can achieve comparable results.

TEST ACCURACIES OF SEMI-SUPERVISED LEARNING ON MNIST, SVHN AND CIFAR-10. THE RESULTS ARE AVERAGED OVER 5 RUNS.

Method	Test Accuracy (%) MNIST N_l=100	SVHN N_l=1000	CIFAR-10 N_l=4000
GAN with feature match [22]	**99.07**	91.89	81.37
CatGAN [23]	98.09	-	80.42
Ladder Networks [24]	98.94	-	79.60
Π-model [14]	-	94.57	83.45
Mean Teacher [16]	-	**94.79**	82.26
VAT [6]	98.64	94.23	85.18
xVAT (Transductive)	98.02	93.99	85.82
xVAT (Inductive)	97.82	94.22	**86.59**

TEST ACCURACIES OF SUPERVISED LEARNING ON CIFAR-10 AND CIFAR-100. THE RESULTS ARE AVERAGED OVER 5 RUNS.

Method	Test Accuracy (%) CIFAR-10	CIFAR-100
Baseline (MLE) [14]	93.24	73.58
Π-model [14]	**94.44**	73.68
Temporal ensembling [14]	94.40	73.70
AT, L_∞(ours)*	93.90	74.04
VAT [6]	94.19	75.02
xAT (Inductive)	93.70	74.62
xVAT (Inductive)	93.88	**75.30**

Speed Comparison

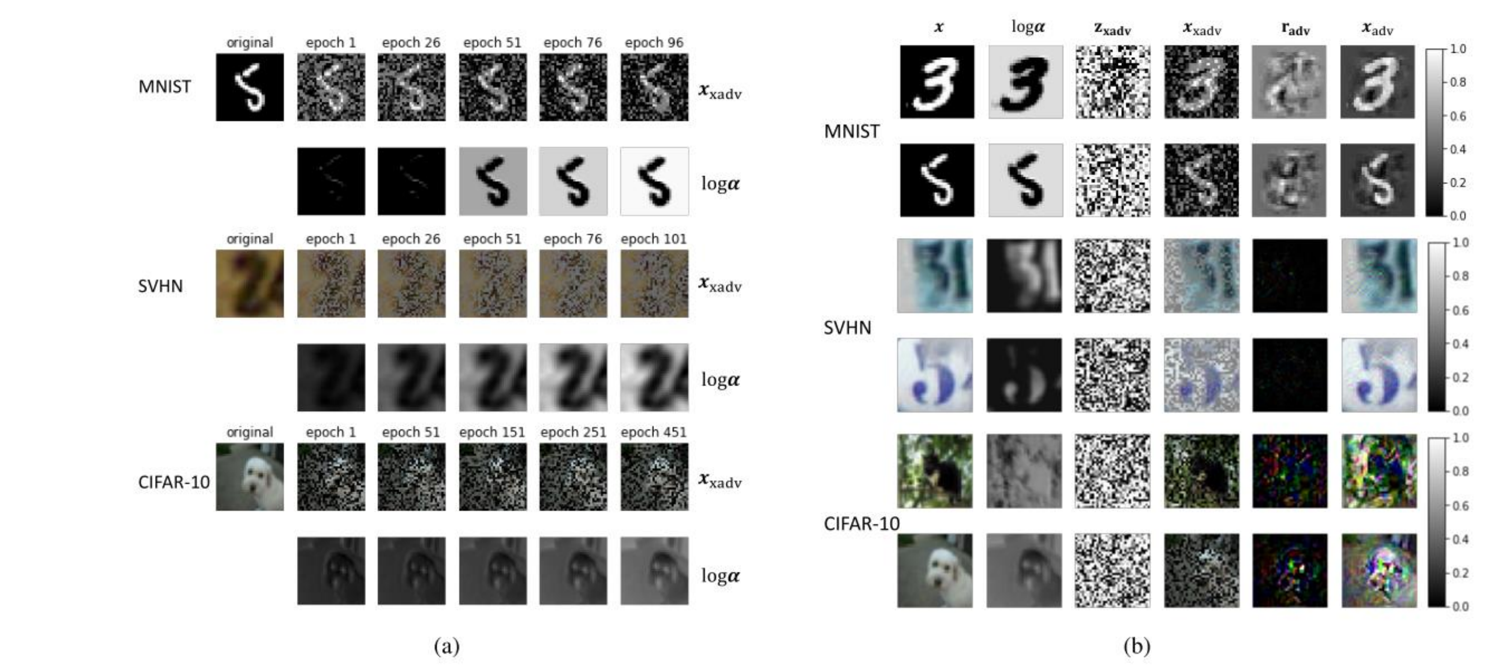
Thanks to the hard concrete reparameterization, the resulting algorithms xAT and xVAT are computationally more efficient than their additive counterparts as the table shows.

THE TRAINING SPEEDS OF VAT AND XVAT ON THE FOUR BENCHMARK DATASETS. THE RESULTS ARE AVERAGED OVER 5 RUNS.

Method	Seconds per epoch MNIST	SVHN	CIFAR-10	CIFAR-100
VAT (ours)*	4.31	54.3	51.3	51.5
xVAT (Transductive)	4.54	36.6	34.1	39.3
xVAT (Inductive)	4.33	35.7	33.6	34.4

Visualization of Multiplicative Perturbations

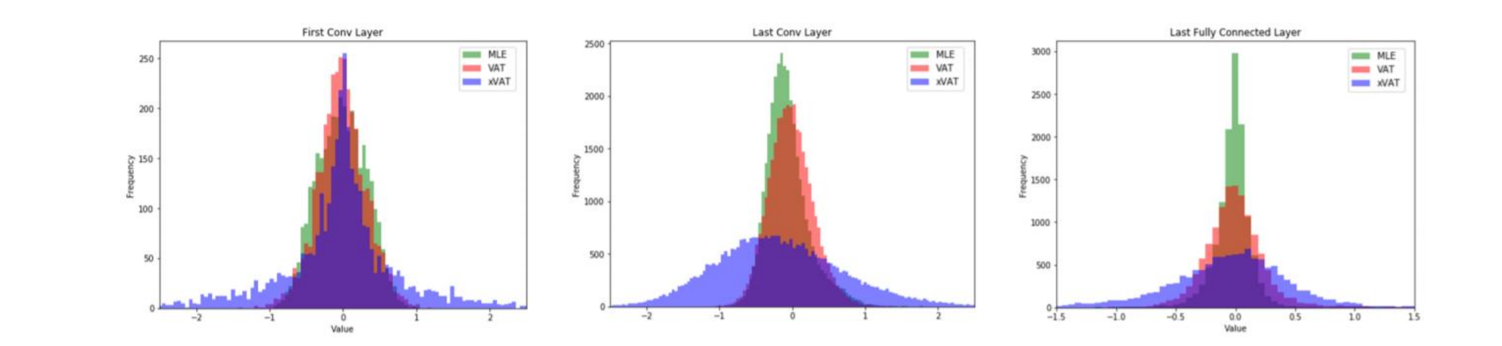
The multiplicative perturbations are
(1) More perceptible (2) More interpretable



(a) (b)

Fig. 5. Visualization of multiplicative perturbations and additive perturbations from xVAT and VAT. (a) The evolution of $\log\boldsymbol{\alpha}$ and $\boldsymbol{x}_{\mathrm{xadv}}$ during the training of xVAT on benchmark datasets. (b) Comparison of multiplicative and additive perturbations on example images from benchmark datasets.

Robustness and Sparsity



(a) First convolutional layer (b) Last convolutional layer (c) Last fully connected layer

Fig. 6. Histograms of the classifier weights learned by MLE, VAT and xVAT on CIFAR-100. The histograms are computed from different CNN layers.